AFHRL-TP-90-78

(2)

AIR FORCE

HUMAN RESOURCES LABORATORY

OBJECT-ORIENTED SIMULATION ENVIRONMENT FOR AIRBASE LOGISTICS

Douglas A. Popken, Capt, USAF

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

November

Interim Technical Paper for Period December 1989 – September 1990

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

BERTRAM W. CREAM, Technical Director
Logistics and Human Factors Division

JAMES C. CLARK, Colonel, USAF
Chief, Logistics and Human Factors Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1990	Interim Paper - Dec 1989 to Sep 1990

4. TITLE AND SUBTITLE

Object-Oriented Simulation Environment for Airbase Logistics

5. FUNDING NUMBERS

PE - 62205F
PR - 1710
TA - 00
WU - 50

6. AUTHOR(S)

Douglas A. Popken

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Logistics and Human Factors Division
Air Force Human Resources Laboratory
Wright-Patterson Air Force Base, Ohio 45433-6503

8. PERFORMING ORGANIZATION REPORT NUMBER

AFHRL-TP-90-78

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Paper presented at the Symposium of the Military Operations Research Society, Annapolis, Maryland, 10-12 June 1990.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Integrated Model Development Environment (IMDE) will address basic difficulties experienced in simulating airbase logistics. These difficulties arise during two major modeling activities: (a) developing the conceptual model, and (b) translating the conceptual model to a particular simulation software implementation. The difficulty of the former activity is largely irreducible, a consequence of the inherent complexity of large, dynamic, and uncertain military systems - such as airbase logistics. However, though the latter activity is also complex, key technologies have the potential to simplify this part of the modeling process. This paper discusses how a synergistic combination of object-oriented programming and data bases, graphical programming, software environments, and powerful computer workstations may provide a breakthrough in simulation modeling capabilities. To further demonstrate concepts, the paper describes an object-oriented simulation prototype of a subset of the airbase logistics domain. The paper also discusses how the IMDE will be designed to meet the needs of airbase logistics modelers with widely differing capabilities and mission requirements. This is to be accomplished by providing different modeling "levels," each with its own user interface, skill requirements, and degree of access to modeling tools. The paper concludes with a discussion of the current status of IMDE development efforts.

14. SUBJECT TERMS

aircraft maintenance
artificial intelligence
data bases
high-level languages
logistics planning
man-computer interface
simulation

15. NUMBER OF PAGES

20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFHRL Technical Paper 90-78 November 1990

OBJECT-ORIENTED SIMULATION ENVIRONMENT FOR AIRBASE LOGISTICS

Douglas A. Popken, Capt, USAF

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

Reviewed by

Wendy B. Campbell
Chief, Logistics Systems Branch

Submitted for publication by

Bertram W. Cream, Technical Director
Logistics and Human Factors Division

Paper presented at the Symposium of the Military Operations Research Society, Annapolis, Maryland, 10-12 June 1990.

SUMMARY

Military systems are often difficult to model because of the inherent complexity of the problem domain. However, although formulation of the *conceptual* model is unavoidably complex, implementation of the model in software need not be. This concept provides the basis for the distinction between what this paper refers to as "essential modeling complexity" and "accidental modeling complexity." *Essential* modeling complexity describes the inherently intractable process of specifying the conceptual model, made all the more difficult when the systems being modeled are large, dynamic, and uncertain. On the other hand, *accidental* modeling complexity is potentially avoidable. It is partially the result of artificial constraints imposed on the modeler by inefficient user-computer interfaces such as text-based programming languages, card-oriented data input, and tabular outputs. It also results from the conceptual "mismatch" between the structure of the systems being modeled and the way software is typically structured.

The Integrated Model Development Environment (IMDE) is a proof-of-concept software prototype being developed to explore the potential for minimizing accidental modeling complexity. It will attempt to exploit several key technologies: object-oriented programming and data bases, graphical programming, software environments, and powerful computer workstations. Object-oriented software technologies are particularly promising because of their ability to provide a close match between the structures of complex, hierarchical "real world" systems and a hierarchical software structure. This concept is demonstrated with an object-oriented simulation prototype of a subset of the airbase logistics domain. While the research effort is primarily geared to airbase logistics modelers, the key enabling technologies have broad application to simulation modeling in general.

PREFACE

The object-oriented simulation environment described in this paper is being developed through an AFHRL research and development contract - Integrated Model Development Environment (IMDE). The prime contractor for IMDE is The Analytical Sciences Corporation (TASC). This contract effort is the major component of a larger research project known as Productivity Improvements in Simulation Modeling (PRISM). The objective of PRISM is to develop advanced simulation technologies for improved logistics capability planning and evaluation. PRISM also currently includes research into object-oriented data management as well as techniques for aggregating simulation processes. The purpose of this paper is to document the current conceptual view of IMDE and how it contributes to PRISM objectives. The "Four-Level Concept" described herein was conceived by Bob Powell of TASC.

I. INTRODUCTION

Simulations are often used to model complex systems. Consequently, simulation approaches have been used to solve problems in a variety of applications, including manufacturing, materials handling, transportation, and defense. Some of the problem areas investigated by military modelers have included battle planning, wartime operations, weapons procurement, force sizing, human resource planning, logistic planning, and national policy analysis (Hughes, 1984). Military systems are often particularly difficult to model because of their frequently large scale and scope, number of interactions and levels of hierarchy, and the uncertainties of wartime dynamics; thus, typical problems are rarely amenable to strictly analytical approaches. For this reason military modelers have often turned to building simulation models for use as decision support aids in planning, analysis, and problem diagnosis.

While simulation has provided military modelers with a means to model complex systems, their use has not necessarily simplified the modeling task. In fact, large-scale military simulation models have tended to be inherently troublesome and complex, for reasons both technical and organizational in nature. Brewer and Shubik (1979, pg. 25) had the following to say about large computer simulation models in the Department of Defense:

> Large computer programs, like large cathedrals, may be built by generations of unnamed workmen using rudimentary plans and developing technologies, subject to the vagaries of changing doctrines and leadership. With large simulations, unfortunately, unlike large cathedrals - where divine guidance was available and fairly simple empirical tests could be performed to find out whether the arches held - the whole structure may collapse or become meaningless without anyone realizing it until many years later. New computer technology has vastly increased our ability to model human affairs, but the abuse of simulation will continue so long as confusion remains between the ability to build large models and the ability to build, understand, and control large models appropriate to the questions being studied.

Recognizing these problems, the Military Operations Research Society organized a series of workshops, SIMTECH-97, to identify and prioritize simulation deficiencies, identify and assess technologies that may ameliorate these deficiencies in a time frame out to 1997, and to recommend follow-on actions (Brady, 1989). Research by the Air Force Human Resources Laboratory (AFHRL) on models used for Air Force logistic planning found many of the existing models "difficult to use, insufficiently documented, and difficult to modify, and to require inordinate amounts of data preparation" (Popken, 1988).

It should be noted, however, that these problems are by no means limited to the realm of *military* simulation models. Simulation models have often been described by analysts and modelers outside of the military as a "tool of last resort" because of their general tendency to be costly, labor intensive, and error prone (Balci & Nance, 1987).

In response to difficulties experienced in modeling Air Force logistics systems, and to the continuing need for simulation models for this purpose, AFHRL has initiated a research thrust known as the Productivity Improvements in Simulation Modeling (PRISM) project. The objective of the PRISM effort is development of a prototype "simulation modeling environment." A proof-of-concept prototype of this environment is being developed with The Analytical Sciences Corporation through the Integrated Model Development Environment (IMDE) contract. The basic idea is to provide an integrated software environment for efficient development, execution, analysis, modification, and overall management of simulation models. The idea is consistent with the concept of a simulation "workbench" outlined in the final report of the Workbench Working Group of SIMTECH-97 (Gilmer & Kameny, 1989).

While the basic idea of simulation environments is not new, the PRISM project would depart from previous efforts in its combined use of the "object-oriented" software paradigm, high-level graphical programming and user interfaces, and powerful computer workstations. The unique qualities of object-oriented software will enable users of the environment to rapidly construct models from modular, reusable component objects. Particular emphasis will also be placed on human engineering and cognitive criteria in the design of the user-computer interface. The synergistic effects of integrating the various features of a graphics-based object-oriented simulation environment within a powerful computer workstation may well lead to breakthroughs in simulation modeling capabilities. While the effort is primarily geared to airbase logistics modelers, the key enabling technologies under current research have broad applicability to simulation modeling in general.

This paper will describe the major functions and features of the simulation modeling environment being developed by AFHRL. The paper will begin with a discussion on modeling complex systems and then show how the major features of the developing simulation modeling environment could reduce the complexity experienced by military modelers or analysts. Following this, the paper will discuss implementation issues relevant to airbase logistics modeling. The paper will conclude with remarks on future development plans.

II. MODELING COMPLEX SYSTEMS

Software construction involves tasks that are *essential*, relating to the composition of a conceptual model of a system, and tasks that are *accidental*, relating to the translation of the conceptual model into a particular computer programming language. Computer programming is inherently complex; in fact, complexity is an essential, not accidental, characteristic of software (Brooks, 1987). Furthermore, as the number of software elements in a computer program increases, program element interactions, and thereby, program complexity, increase exponentially. Being a type of software, simulation models have these same basic characteristics. By implication then, the complexity of simulation software increases exponentially with the size of the model.

In typical military problem domains, the systems being modeled are often large, complex, and involve many possible states and interrelationships; thus, the abstract conceptual model itself invariably tends to be highly complex. This "essential modeling complexity"[1] is unavoidable. On the other hand, "accidental modeling complexity" is potentially avoidable. This term refers to the gap between conceptual models and implementation of those models in computer programming languages. Part of this gap results from the characteristics of existing user-computer interfaces. Examples include text based computer programming languages, card-oriented data entry procedures, and output in the form of voluminous printouts of columnar data.

Improper specification of the conceptual model can also lead to a type of accidental modeling complexity. This is the result when the modeler fails to represent only the *essential* aspects of a physical system in the conceptual model. Part of the difficulty lies in the inherent mismatch between the structures of "real world" systems and the typical structures of the software models of these systems. This mismatch is reflected in the inconsistency between object-oriented and functional decomposition analyses (Coad & Yourdon, 1990), two of the structured analysis techniques available for analyzing the physical domain . This type of accidental modeling complexity can also result from an inability to properly limit the scope of a model to the immediate problem.

As modeling is as much an art as a science, accidental modeling complexity is hard to measure. In the case of military simulation models, what can be said is that accidental modeling complexity is a frequent byproduct of what shall be referred to here as the

[1] Here we will borrow the terms "essential" and "accidental" from Brooks, but extend their original definitions to apply to the process of mapping physical systems to computer simulation models.

"standing model paradigm." The term "standing model" refers to a computer model, typically large scale, that is designed to be used over time to address a fixed set of repetitive problems. The basic idea behind using such a model is to avoid having to recreate a new model for every problem; only the input data and certain parameters are adjusted. A large number of simulation models used by the military are operated in this way (Joint Staff (J-8), 1989). Such an approach can be viewed as a practical response to a decision-making environment constrained by time and by the computer technology available in the 1960's and 70's. The main drawback to this approach is the lack of capability to focus on only the essential elements of a problem. "Instead of trying to determine in advance which were the important variables, their [military analysts] brute-force method treated all variables as important and assumed that the computer would overpower nature. As usual, nature won, overwhelming the computer and the analyst with her complexity." (Hughes, 1984, pg. 22)

While there have been major advances in computer technology since the 1960's, particularly in computer hardware, the standing model paradigm still dominates DoD. This is largely a result of bureaucratic tendencies towards caution (Kornell, 1987) and the relatively slow rate at which new technology enters nonoperational military domains. New complexity-reducing modeling paradigms are now needed to exploit the potential of emerging computer technologies

The PRISM effort attempts to reduce modeling complexity, and thereby increase productivity, by minimizing accidental modeling complexity. To understand how this can be achieved, it is useful to initially discuss the environment's proposed features under the categories of its primary enabling technologies: object-oriented programming, graphical programming, object-oriented data bases, software environments, and workstations. No one of these features will likely bring about an order of magnitude increase in modeling productivity; however, their thoughtful integration would make the potential for this very real.

III. ENABLING TECHNOLOGIES

Object-oriented Programming

The basic argument for object-oriented programming is its ability to deal with complexity. Software, of which simulation models are one type, is inherently and often arbitrarily complex (Booch, 1990). As can easily be the case with the "standing models" discussed above, the complexity of software can exceed the human intellectual capacity.

Object-oriented software incorporates the concepts of modularity and hierarchical structure. Modularity is accomplished through the encapsulation of procedures and data into software "objects" that communicate only through passing "messages" among themselves. Objects represent "instances" of some particular class of objects. Relationships between classes can be defined through a class hierarchy, whereby specialization and detail increase as one traverses from the root to the leaves in the hierarchical tree (Figure 1). Only those elements essential to defining unique state and behavioral characteristics are implemented in a given class. All other characteristics are "inherited" from higher level classes in the hierarchical tree.

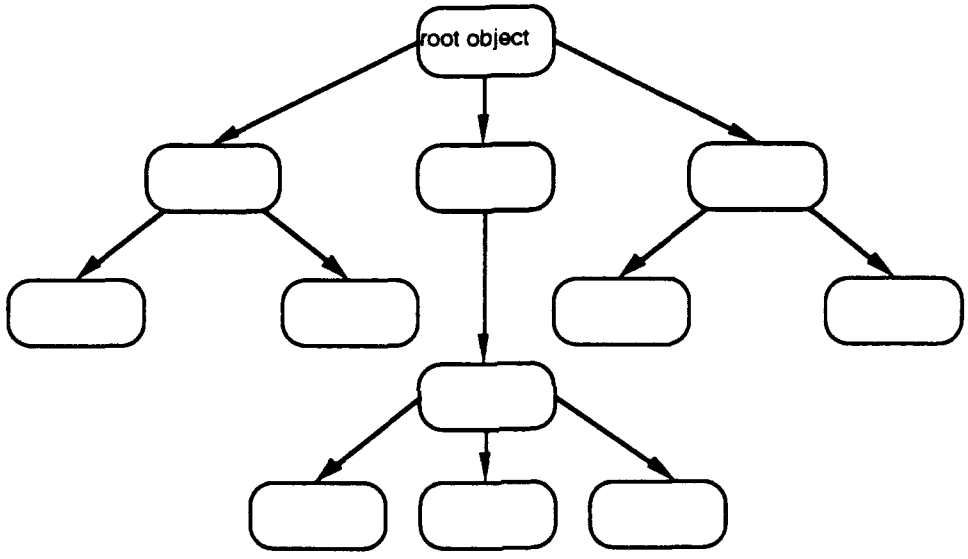

Figure 1. A Hierarchy of Object Classes

The object-oriented paradigm reduces the complexity of simulation models through encapsulation and hierarchical structuring of components. These characteristics are particularly relevant to simulations for several reasons. First, simulations usually model some physical system in the "real world". Such systems can be viewed in terms of two orthogonal hierarchies: 1) objects and their component parts, and 2) classes of objects and their subclasses. In object-oriented simulation, the physical objects and their associated hierarchies in the modeled system will often have direct counterparts in software objects and hierarchies. An understanding of relationships between objects in the physical domain can then be more easily translated into relationships within the software domain. This

mapping characteristic allows the physical world, the conceptual model, and the software implementation to use a consistent organizing framework.

Second, in the real world, complex hierarchic systems tend to be composed of relatively few types of subsystems. The perceived complexity results largely from merely arranging the subsystems in different ways. Object-oriented simulations of these physical systems are similar in this respect. This feature, along with inheritance, allows a developer to focus on essential details of the model.

Third, modularity localizes the effects of any necessary modification of the simulation software. Modification of simulation models may be frequent, resulting from changes in the application domain or in the way the domain is being modeled. Modularity allows the developer to concentrate on only the essential details of modification.

A number of programming languages have been developed to directly support the object-oriented paradigm, including Simula, SmallTalk-80, C++, Objective-C, MODSIM II, and various object-oriented extensions to the LISP language such as Flavors and Scheme. Though often described as an "object-oriented" language, the Ada language is not part of this group because it does not support classes or inheritance.

Selection of a particular object-oriented language involves tradeoffs among criteria such as flexibility, expressive power, run-time performance, and portability. C++ and MODSIM II are being considered as the underlying language for the prototype Integrated Model Development Environment (IMDE). Both of these languages have good run-time performance, wide portability, and the ability to interface with traditional programming languages such as C. C++ is becoming the predominant language in commercial object-oriented applications for similar reasons. For IMDE this would mean greater opportunities for incorporating or interfacing with commercial off-the-shelf software components. On the other hand, MODSIM II is attractive because of its built-in simulation class libraries.

Graphical Programming

Graphical programming refers to computer programming through manipulation of symbolic images on a computer screen. The primary motivation of graphical programming in the PRISM effort is simplification of the user-computer interaction rather than animated portrayal of the simulation. Most readers of this paper are already familiar with the simplest example of graphical interaction, selection of options from a "menu" of choices. The IMDE effort will go much further than this to include representation of complex relationships between simulation objects. For example, relationships could be defined in a temporal sense, as would be the case for a network of simulated activities, or in the sense

of object aggregations, whereby objects are identified as part of some larger group[1] (Figure 2). Modelers would manipulate on-screen objects to define their characteristics and interrelationships. The graphical models will be translated automatically into executable simulation programs. The SmallTalk-80 programming environment is a current example of an object-oriented system capable of these types of interaction.

Graphical representation of complex modeling relationships provides rapid and straightforward feedback on the status of model construction. The gap between the conceptual model and its implementation is then greatly reduced, particularly when the graphical programming interface is consistent with the conceptual model . A number of activity network-based versions of this approach already exist in commercial form for standard simulation languages (Bell & O'Keefe, 1987). Initial research has demonstrated the feasibility of using "activity cycle diagrams" (Ozden, 1990) as a graphical representation of object-oriented simulation models.

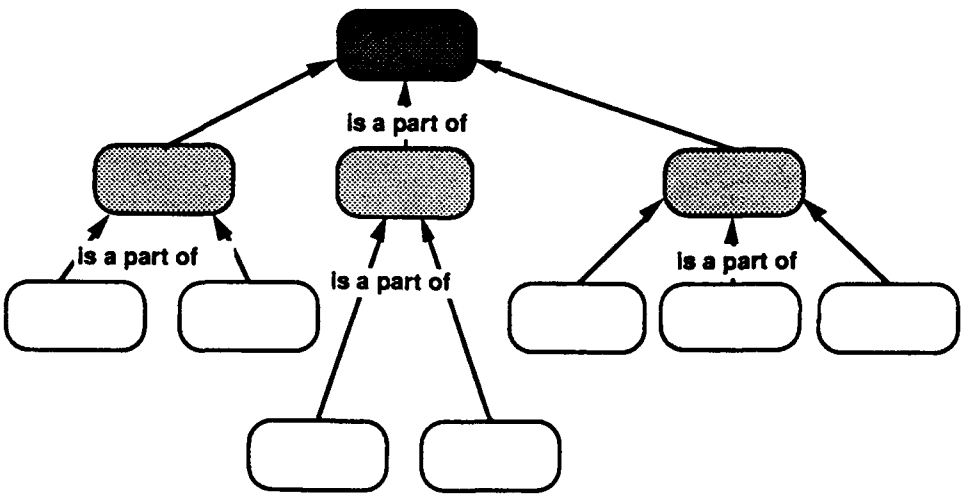

Figure 2. A Graphical Representation of an Object Aggregation Hierarchy

[1] This type of object relationship, often referred to as "is a part of," is in contrast to the "is a kind of" relationship defined by the class hierarchies discussed in the previous section.

Object-oriented Data Bases

An object-oriented data base (see, for example, Zdonik & Maier, 1990) provides persistence for simulation objects. Without it, objects exist as objects only as long as the simulation program is running; they would have to be either manually or automatically reconstructed at each execution from "flattened" data structures. Because of the resulting "mismatch," run-time efficiency would be degraded; complex data retrieval statements would be needed to translate, say, relational data, to object form and vice versa. Further, system management of data would be more difficult, particularly if objects are being shared between users or platforms.

An object-oriented data base contains objects that may vary in size and structure to nearly any arbitrary degree. Complex structures and compound objects can be represented directly. Objects can simultaneously hold text, numbers, and bit-map representations of data.

These features of object-oriented data bases will help support rapid construction of simulation models in IMDE. A modeler will be able to browse through a data base hierarchy of validated objects (including compound objects), select those most appropriate to his problem, and iteratively build and refine a model composed of these objects. This implies that prior modeling knowledge and expertise are captured for reuse at the object and at the compound object level. This would be a distinct improvement over the "standing model paradigm," where the entire model must be retained as the atomic unit of reusable experience. The resulting potential for complexity reduction is clear to anyone who has attempted to modify the source code of a large computer program written by another programmer.

Commercial object-oriented data bases are only now reaching adequate levels of maturity and robustness. A number of vendors now have good quality systems available. It has not been determined whether IMDE will use a commercial off-the-shelf object-oriented data base or a custom-developed object-oriented data base.

Software Environments

A software environment provides an organizing framework for the software tools needed to implement the enabling technologies discussed above. The environment will provide an integrated set of tools for development, execution, analysis, modification, and overall management of simulation models. The object-oriented data base at the core of the

environment will work through the environment to provide an integrated repository for all simulation model and project information. The environment will include a consistent and easy-to-use graphical interface.

Higher level features of the IMDE include its support for rapid prototyping and its emphasis on the overall modeling process. In general, rapid prototyping is used to address the fact that system requirements are never fully known at the outset of a development effort. In modeling airbase logistics, as well as other modeling efforts, the end user of simulation derived data may specify the desired information in only the most general terms. Rapid prototyping allows iterative specification of requirements. Each iteration incorporates feedback on the performance and desirability of the features of each successive prototype.

IMDE is being designed to support rapid prototyping and incremental development of simulation models through the graphical and object-oriented programming techniques described previously in this paper. Graphical programming allows immediate visual feedback on the current state of development and direct evaluation of the current conceptual model of the simulated system. The object-oriented paradigm supports rapid prototyping by providing standardized, pretested software modules in the form of objects. New classes are created quickly as extensions of existing classes. The model builder can then confine his attention to integration of subassemblies, allowing rapid construction or modification of larger model assemblies.

Rapid prototyping reduces complexity by dividing system specification into manageable portions, where each portion corresponds to an iteration in the development process. To paraphrase a basic mathematical principle, since the complexity of a simulation model increases in a greater than linear (convex) fashion with its size, the "sum of the complexities" of individual development iterations is much less than would be the complexity of developing a total system in a single cycle.

An additional high-level characteristic of the environment will be its emphasis on the modeling process rather than on the models alone. That is, IMDE will be designed to reinforce the notion that the simulation model is but one component of the analysis process. The simulation model "merely" generates the experimental data. The data must then be statistically analyzed to determine whether some statistical hypothesis regarding the modeled system has been reasonably satisfied. This process typically involves numerous iterations.

IMDE is also being designed to accommodate the varying modeling needs of a potentially diverse set of users; however, this topic will be discussed more fully in a following section "The Four-Level User Concept".

Workstations

Modern computer workstations provide the computational power needed to drive object-oriented simulations with reasonable ease. High-speed machines are desirable due to the relatively slow performance of object-oriented software when compared to traditional programming languages. Graphics generation requirements place additional burdens on the system. Workstations also provide support for refined networking, access control, and security features that are not found at the PC hardware level. The prototype IMDE will reside on a SUN-4 workstation. However, portability across other workstations will be a primary design goal.

IV. IMDE USER CONSIDERATIONS

Modeling Airbase Logistics

The initial target application domain for IMDE is airbase logistics modeling. This domain encompasses the logistics processes that directly support aircraft sortie generation at an operational airbase. This would include such things as aircraft maintenance, parts supply, and munitions loading. Logistics models of this type have been used for studies in aircraft acquisition planning, maintenance manpower allocation, and theatre-level supply redistribution.

The specific problem scenarios modeled in this domain are diverse. A typical example in acquisition planning might be determining the effects of introducing a new version of an aircraft component part with a greater mean time between failure and/or a shorter average repair time. In a simulation model of this problem, sorties are generated and aircraft return to base with parts (including the one in question) having failed according to some probability distribution. The aircraft then goes through the maintenance processes necessary to return it to flying status and additional sorties. Thus, all other parameters held constant, a relationship between component part characteristics and sortie generation rates can be statistically estimated. Similarly, all manner of proposed aircraft characteristics, components, processes, and policies can be tested in relation to their potential effects on sortie generation rates.

The airbase logistics problem domain can be decomposed using the techniques of object-oriented analysis (Coad & Yourdon, 1990). Figure 3 illustrates representative features of one possible decomposition. A prototype object-oriented simulation using a more complete version of this hierarchy has been developed by the author in the SMALLTALK-80 programming environment.

At the top of the object hierarchy depicted in Figure 3 is the generic class, **Object**. Classes at lower levels of the hierarchy show greater specialization. Under the object-oriented paradigm new classes may be created as a specialization of an existing class. For example, to introduce a new type of aircraft to the simulation domain, a specialization of the existing class, **Aircraft,** is created. The new class will inherit state and behavior information from the generic **Aircraft** class. Only information unique to the new class of aircraft must then be entered into the system.

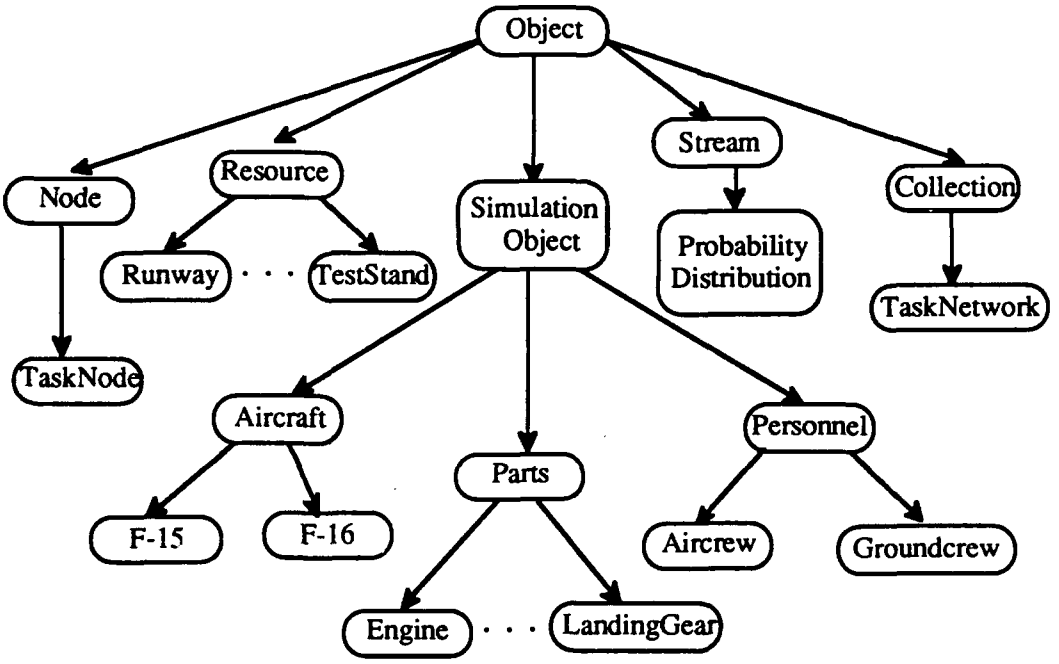

Figure 3. Representative Object Classes in Airbase Logistics Simulation

In object-oriented simulation modeling there are typically classes representing domain specific objects, such as **Aircraft**, and general classes of more abstract objects that support the desired interactions of the domain objects and the simulation process itself. For

example, Figure 3 includes the class, **ProbabilityDistribution**. This class responds to messages that access random variables, parameters, moments, etc. Subclasses of **ProbabilityDistribution** would implement behaviors for specific probability density functions. Also note the classes **TaskNode** and **TaskNetwork**. These classes can be used to implement probabilistic sequencing of activities. Other classes of abstract objects are used to implement basic simulation processes such as queuing and event scheduling.

The Four-Level User Concept

As is the case with the models themselves, the organizations using airbase logistics models are diverse, having a wide range of missions, requirements, and resources. Personnel may have widely varying levels of experience and education. At one end of the personnel spectrum are pure model users: those with no formal education in simulation, statistics, or systems modeling techniques, but who have been trained to perform simple parametric studies with a given simulation model. At the other end of the spectrum are model developer/users: people with the requisite background to use models, build models, or given enough time, modify existing simulation models. Because the group of potential users is so diverse, an object-oriented simulation environment would not automatically be suitable for all potential using organizations or users, despite its advantages.

The PRISM effort will attempt to maximize the potential user base through a "Four Level" user concept (Figure 4). Under this concept, the amount of modeling expertise required of the user depends on the degree of access to system functionality and capabilities desired by that user. Higher levels allow greater access. To maintain model integrity, access to higher levels in the environment will be limited to only those with the necessary expertise. At each of the four levels, the user interface will be consistent with the user's view of his tasks, goals, objectives, and expertise.

Level I, the "User Level," will require the least amount of expertise. Here, model users will be able to perform simple parametric studies with predefined simulation models and statistical analysis tools. This level will be designed to encourage "learning by doing." The Level I user will not be given access to higher levels.

Level II, the "Analyst Level," is where specific objects may be selected from an object-oriented component library contained within the object-oriented data base. Specific models are built, tested, and then compiled for distribution to Level I. No access is provided to higher levels.

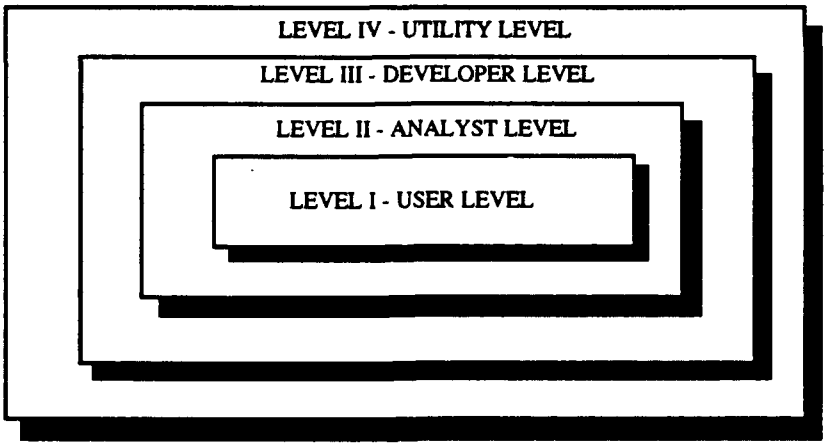

Figure 4. Four-Level Concept

Level III, the "Developer Level," will allow building, testing, and validating of new simulation objects for lower levels. Using the object-oriented paradigm, new objects are defined as specializations of already existing object classes. Access to object procedures, or "methods" as they are known, will allow model interactions and relationships to be altered. The Level III user will be given limited access to Level IV.

Level IV is the "Utility Level." Here, access is provided to the underlying software environment. General tools such as linkers, compilers, code generators, and others are available to qualified people.

V. PRESENT STATUS AND FUTURE PLANS

Preliminary investigation of object-oriented data bases, graphical programming, and object-oriented design of the airbase logistics domain has demonstrated the viability of the PRISM concept. In March of 1990 a three-year effort was initiated to develop a proof-of-concept Integrated Model Development Environment. The objective of Phase I of this contract effort is to produce a detailed description of system requirements. Particular emphasis is to be placed on analyzing high risk areas, of which there are several. Foremost among these is in the design of the user-computer interface. The interface must simultaneously be both powerful and easy to use. It must also be designed to support, and to a degree, enforce good modeling practice. Note also that the Four-Level concept

described above implies that IMDE will have not just one, but *four* different user interfaces.

The design of the object-oriented data base and its interface with the rest of the system is another high risk item. The data base must be designed such that data management is both efficient and transparent to the user. That is, users need not concern themselves with whether data objects reside in memory or in secondary storage.

A number of other design considerations will center around modeling issues. These include but are not limited to providing for sound statistical analysis of output data without simultaneously making the process overly complex or cumbersome, providing the analyst with the ability to model at variable levels of detail, and reconciling the "information hiding" characteristic of objects with the need to make modeling assumptions known to the user. Clearly, an object-oriented simulation environment is rich with potential research issues.

IMDE is being developed using rapid prototyping techniques. Early prototypes will concentrate on the modeling interface. A demonstration of a preliminary prototype of major portions of the environment is scheduled for September 1991.

REFERENCES

Balci, O. & Nance, R. (1987). Simulation support: prototyping the automation based paradigm. In Thesen, A., Grant, H., and Kelton, W. (Eds.). Proceedings of the 1987 Winter Simulation Conference (pp. 495-502). Atlanta, GA: Society for Computer Simulation.

Bell, P. & O'Keefe, R. (1987). Visual interactive simulation - history, recent developments, and major issues. Simulation, 49(3), 109-116.

Booch, G. (1990, January). Object-oriented design. Notes from tutorial course presented at Seminars and Conferences in Object-oriented Programming (SCOOP), Santa Clara, CA.

Brady, E. (1989). SIMTECH-97: an overview. Phalanx, 22(1), 19.

Brewer, G. & Shubik, M. (1979). The war game. Cambridge, MA: Harvard University Press.

Brooks, F., Jr. (1987). No silver bullet: essence and accidents of software engineering. Computer, 20(4), 10-19.

Coad, P. & Yourdon, E. (1990). Object-oriented analysis. Englewood Cliffs, NJ: Prentice-Hall .

Gilmer, J. & Kameny, I. (1989). SIMTECH-97 report of the workbench working group. Phalanx, 22(4), 30-33.

Hughes, W. (Ed.) (1984). Military modeling. Alexandria, VA: Military Operations Research Society.

Joint Staff - Force Structure Resource and Assessment Directorate (J-8) (1989). Catalog of wargaming and military simulation models (AD-A213970). Washington, D.C.

Kornell, J. (1987). Reflections on using knowledge based systems for military simulations. Simulation, 48(4), 144-148.

Ozden, M. (1990). Graphical programming of simulation models in an object-oriented environment. Paper presented at a meeting of the Society for Computer Simulation, Nuremberg, FRG.

Popken, D. (1988). Productivity improvements in simulation modeling: concepts and motivations (AFHRL-TP-88-31). Wright-Patterson AFB, OH: Logistics and Human Factors Division, Air Force Human Resources Laboratory.

Zdonik, S. & Maier, D. (Eds) (1990). Readings in object-oriented database systems. San Mateo, CA: Morgan Kaufmann Publishers.